1):host

In bracket we can give selector for component like some class, id etc

It applies style to component.

a):host

:host {

border: 2px solid red;

}

Generated selector-

[\_nghost-c1] {

border: 2px solid red;

}

2)

:host(.componentClass) {

color: violet;

}

Generated style-

.componentClass[\_nghost-c1] {

color: violet;

}

3)

:host(.componentClass).componenetClass2 {

color: violet;

}

Generated styles-

.componentClass.componenetClass2[\_nghost-c1] {

color: violet;

}

4)

:host(.class1)p.class2 {

    display: inline-block;

    border: 5px solid black;

}

Generated style-

.class1[\_nghost-tdr-c16] p.class2[\_ngcontent-tdr-c16] {

    display: inline-block;

    border: 5px solid black;

}

5)

:host(.componentClass)#id1.componenetClass2 {

color: violet;

}

Or

:host(.componentClass#id1.componenetClass2) {

color: black;

}

Generated styles-

.componentClass#id1.componenetClass2[\_nghost-c1] {

color: violet;

}

6)

:host(.componentClass) .classInsideComponent {

color: violet;

}

Generated style-

.componentClass[\_nghost-c1] .classInsideComponent[\_ngcontent-c1] {

color: violet;

}

7):host p

:host p {

color: black;

}

Generated style-

[\_nghost-c1] p[\_ngcontent-c1] {

color: black;

}

8)

:host(.componentClass .classInsideComponent) p {

color: violet;

}

Generated style-

.componentClass[\_nghost-c1] .classInsideComponent[\_ngcontent-c1] p[\_ngcontent-c1] {

color: violet;

}

2)::ng-deep h1, /deep/ h1, >>> h1

Whatever we write after ::ng-deep it will be decared as global style. If you do not want to have any filter, i.e we do not want to declare global style, then that filter comes before ::ngdeep

If we do not have anything before ::ng-deep then, the selector after ::ng-deep will be global (ex-1)selector.

If we have anything before ::ng-deep then(selector1), then selector(selector 2) after ::ng-deep will be applied only to childs of elements selected by selector1.

1)::ng-deep

::ng-deep p {

color: green;

}

Generated style-

p {

color: green;

}

We are not using anything before ::ng-deep, hence it is style is defined for all p tages.

2):host ::ng-deep

:host ::ng-deep p {

color: green;

}

Generated style-

[\_nghost-c0] p {

color: green;

}

useful in styling the projected components.

Here we are :host before ::ng-deep, style is applied to all p tags that are in component.

3):host\_context

It selects the host element depending upon, class or id applied to its parent. But we further configure it by specifying what class needs to be on component.

In bracket we give selector for parent element and after bracket we give selector for component(if we are not using space after bracket, ex-4) or selector for element inside component(ex-5)

1)

:host-context(.parentClass) {

color: red;

}

Generated style-

.parentClass[\_nghost-c1], .parentClass [\_nghost-c1] {

color: red;

}

2)

:host-context(.parentClass).componentClass {

color: red;

}

Generated syles-

.parentClass.componentClass[\_nghost-c1],

.parentClass .componentClass[\_nghost-c1] {

color: red;

}

3)

:host-context(.parentClass)p {

color: red;

}

Generated style-

.parentClassp[\_nghost-c1],

.parentClass p[\_nghost-c1] {

color: red;

}

4)

:host-context(.parentClass) .elementInisdeComponentClass {

color: red;

}

Generated style-

.parentClass[\_nghost-c1] . elementInisdeComponentClass[\_ngcontent-c1],

.parentClass [\_nghost-c1] .elementInisdeComponentClass[\_ngcontent-c1] {

color: red;

}

5)

:host-context(.sood#sumit) ::ng-deep h1 {

color: yellow;

}

Note- this is combination of 2 selectors. Evaluate from left to right. First evaluate, :host-context, then evaluate ::ng-deep

Generated style-

.sood#sumit[\_nghost-c1] h1, .sood#sumit [\_nghost-c1] h1 {

color: yellow;

}

6)

:host-context(.parentClass).ComponentClass ::ng-deep h1 {

color: yellow;

}

Generated style-

.parentClass.ComponentClass[\_nghost-c1] h1,

.parentClass .ComponentClass[\_nghost-c1] h1 {

color: yellow;

}

7)

:host-context(.parentClass) .insideComponentClass ::ng-deep h1 {

color: yellow;

}

Generated style-

.parentClass[\_nghost-c1] .insideComponentClass[\_ngcontent-c1] h1,

.parentClass [\_nghost-c1] .insideComponentClass[\_ngcontent-c1] h1 {

color: yellow;

}